ARL-TN-1287 ● Dec 2025

Science of Intelligent Systems Research Operations

by Jeffrey Twigg, Francisco Serna, Emma Holmes, Kiana Bronder, Benjamin Dossett, Sandeep Alankar, and Julie Foresta

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TN-1287 ● DEC 2025

Science of Intelligent Systems Research Operations

Jeffrey Twigg, Kiana Bronder, Benjamin Dossett, Sandeep Alankar, and Julie Foresta
DEVCOM Army Research Laboratory

Francisco Serna and Emma Holmes
Johns Hopkins University Applied Physics Laboratory

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
December 2025	Technical Note	10/15/2024	9/29/2025

4. TITLE AND SUBTITLE
Science of Intelligent Systems Research Operations

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
N00024-22-D-6404		
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Jeffrey Twigg, Francisco Serna, Emma Holmes, Kiana Bronder, Benjamin Dossett, Sandeep Alankar, and Julie Foresta

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DEVCOM Army Research Laboratory
ATTN: FCDD-RLA-JB
Adelphi, MD 20783

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TN-1287

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCIDs: Jeffrey Twigg, 0000-0003-0423-1095; Benjamin Dossett, 0000-0003-4831-1943; Emma Holmes, 0009-0008-3986-1381

14. ABSTRACT
During the development of Phoenix R1, the scale of contribution and robot experiments grew. Although ARL developed some continuous integration and continuous deployment tools, these increasing scales made maintaining and guaranteeing the capabilities of robots difficult. In this technical note, we detail our goals, scope, and requirements for the reliable deployment of robots. We also provide further context for the requirements by describing anticipated users and use cases. Finally, we outline a possible approach to achieving the requirements.

15. SUBJECT TERMS
operations, autonomous robots, development, artificial intelligence, AI, Military Information Sciences

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	27

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Jeffrey Twigg	(520) 691-5861

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Contents

List of Tables

1. Introduction

The Science of Intelligent Systems at the U.S. Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory (ARL) has a leading role in the development and transition of autonomous ground navigation capabilities. It is the leader of several programs with academic and industry partners at "Technology Readiness Level" 6.1 and 6.2. Since 2015, the scales (i.e., number of collaborators, transition partners, number of robots, and number of diverse geographic test sites) at which ARL conducts robotic experiments have grown. Simultaneously, a major confederation of software, Phoenix, has undergone a major revision from revision 1 (R1) to revision 2 (R2). In this report, we examine the users and use cases in which software tools are required to streamline the operations of software development and testing on ground robotic systems. We envision this set of tools will be similar to Development Operations (DevOps); however, we emphasize development of autonomy software and deployment on a limited set of ground robots with data collection and training. We call this new set of operations "Research Operations" or "research ops."

1.1 Goals

Our research goals are outlined below.

- Create a streamlined process for research ops that enables the following:
  - Using a complex codebase across multiple use cases
  - Accelerating research development across many contributors
  - Managing a complex codebase across multiple institutions
  - The process must engage software development at the rate that it occurs. Any tooling that slows down software development will be ignored. Development speed has been a key factor in success, and a new research ops approach should not undercut this.
- The research ops are similar to the DevOps except for the following:
  - There are more contributors with somewhat aligned goals who contribute over different timetables.
  - Code is integrated for knowledge products or functioning robots, and, as a result, we will need to identify functional versus research code.
  - The ecosystem for developing software for autonomous ground robots. These systems may or may not be connected to a network to suit a mission requirement.

1.2 Scope

The scope of this effort is solely research ops (i.e., development and early experimentation, exclusionary of autonomy or hardware integration). This includes tooling for code development, code deployment, data management, and documentation.

- Infrastructure and processes that support autonomy development, experimentation, transition, and management
- Continuous integration and continuous deployment (CI/CD) GitLab, Stack catalog (i.e., components and how they fit together), and documentation
- Improving robotic performance and developing tools to introspect robotic performance is not in the scope of this report. Although there is an overlap between the physical constraints of the platforms and their related configurations, the best ways to display, understand, and resolve these performance issues are subjects of open research.

1.3 Context of the Deploying Autonomy Software in 2025 and Beyond

After using our code Stack that has been based in the Robot Operating System (ROS) 1, Phoenix R1, for approximately 5 years, we have more academic collaborators and robots, and we test in more locations. However, we continue to focus on a small team of maintainers to manage merge requests (MRs) into a main branch. The MRs will be scaled to the number of collaborators. Deployment will be more difficult as the number of robotic platforms and their complexity increase. Notes about those scales are as follows:

- Support for 2 days to 2 months of field experimentation
- 4 TB per hour for each large, multisensor robot
- 1 TB a day for one small robot, such as a Jackal
- 5 Warthogs, 6 Huskies, 10 Jackals
- Quadrupedal robots (Ghost Robotics Vision 60)
- Potentially hundreds of internal and external collaborators

1.4 Autonomy Stack Users

An appropriate approach to satisfying these requirements is to create a taxonomy of the users of Phoenix R2. These user types impose implicit constraints on the approach and provide a foundation for the requirements we will describe. Each user type is defined by a level of expertise and objectives to accomplish by using the Phoenix R2 Stack.

Users assume different roles, as listed below, with the following objectives:

Developers: Collaborative research agreements (academic institutions, government individuals, etc.)

- Academic (publish papers, release code)
    - 2-year projects mostly done by (graduate) students and professors/principal investigators
    - This role is likely thinking longer term, across multiple programs of students with some overarching product goal
    - Low robustness: little testing is performed on added capabilities; therefore, it is unlikely for this software to be directly adopted into the main branch
- ARL Researcher (create knowledge product, transition a capability)
    - Test and develop academic approaches
    - Develop and test capabilities from components (requires coordination with maintainers)
    - Develop feature branch development and component testing --> system deployment

Maintainers: Test and approve MRs for a coherent codebase that are closely affiliated with ARL's Science of Intelligent Systems Division (SISD).

- Conduct code review and regression testing for a coherent, stable codebase to complement ARL hardware and test environments
- Knowledge of configuration and dependency as well as test methodology
- Approve MRs

Operators: (Likely, internal to the Science of Intelligent Systems Division [SISD] at ARL.) Use the Phoenix R2 Stack and reconfigure it for research or demonstration. Operators are expert users who work closely with maintainers and developers.

- Use Stack as baseline for some other feature they are testing
- View our Stack as a product to be used and reconfigured
- Use software on well-tested platforms, sensors, and software deployment
- Physically interact with the Stack and platform (hardware testing) in new environments
- Engage with configuration management tools, not modifying the Stack (no recompile)

Consumer: (Likely, external to SISD/ARL.) Use the Autonomy Stack as the foundation for a capability. The consumers are not expert users.

- Will not interact with the Stack, they just need any Autonomy Stack to run known and tested capabilities
- May use different platforms, sensors, and software deployments

Transitioner: Exist at the program management level and external to SISD and ARL (e.g., DEVCOM Ground Vehicle Systems Center [GVSC] or Defense Advanced Research Projects Agency [DARPA])

- Goal: cross-linking systems (e.g., pulling specific components for specific uses)
- May draw specific components or capabilities

Managers: Exist at the program management level and internal to SISD and ARL

- Goal: provide big picture of system usage and functionality (Controlled Unclassified Information [CUI], licensing, contributions)

Autonomy Stack users are compared in Table 1.

Table 1. Phoenix autonomy Stack users.

	Needs (input)	Activity	Provides	Collaborate through
Developer	• Documentation • Datasets	• Run experiments • Pulling baseline (with data) • Test cards • Documentation	• New features • New component • Utility software	• Datasets • Metrics
Maintainer	• Documentation • Test cards	• Tests with broad configs • Preflight checks • Test cards • Documentation	• Integration of components • Configuration changes • Infrastructure and testing development	• Datasets • Metrics • Bug/issue reports
Operator	• Documentation • Controlled deployment tools	• "Preflight check" • Expose problems • Validation	• Intra-configuration changes • Hardware changes • Environmental changes	• Bug/issue reports • After-action report (AAR) • Datasets
Consumer	• Documentation	• Preflight check • Expose problems • Validation	• External-configuration changes	• Bug/issue reports • AARs
Transition	• Metrics • Datasets	• Validation steps • Licensing • Documentation	• Extraction/ generalization of approaches • Refactoring	• Requirements • Documentation
Manager	• Documentation • Metrics	• Licensing	• Requirements	• Requirements • Metrics • Architecture • Provenance • AARs

Each user type consumes input information to conduct an activity to provide a product. Users work with each other through the objects in the "Collaborate through" column. Definitions of terms in Table 1 are as follows:

- Test card: A policy for preparing and executing a test
- Metrics: Measurements of autonomous platform performance
- Config: A set of parameters and calibrations for a robotic experiment

- Preflight check: A set of tools for evaluating an autonomous platform before executing a test
- AAR: After-Action Report about the results of a series of experiments

1.5 Requirements

The following requirements are proposed to achieve the described goals. The research ops system shall enable

1) A continuous, **baseline-usable state**, which is the "main" branch of Phoenix R2 consisting of the following:
   a) Active documentation for all users to develop, test, instantiate, troubleshoot, and understand the system in simulation and on robots.
   b) A common testing scheme that compares with the baseline to ensure a usable state across a specific common set of use cases (platform, environment, scenario, dependencies) that consist of the following:
      i) unit tests
      ii) simulation tests
      iii) on-robot tests
   c) Reviews/quality checks for updates/upgrades/bugfixes to baseline that ensure
      i) code contributions to the baseline meet the standard of code quality and provide adequate documentation
      ii) dependency changes are managed
      iii) bugs in the software are documented and addressed
      iv) platform deployments are successful

This requirement establishes the idea of a main branch of Phoenix R2 that is reliable as 1) time advances and 2) software is added. Documentation should be updated routinely for all Autonomy Stack users defined above to ensure that everyone has instructions on how to set up and use the system as well as learn about current capabilities. We imagine that various unit tests, simulations, dependencies, test data (rosbags), and other components needed for testing a common set of use cases will be maintained as part of the baseline. Ideally, the baseline can be configured in "n" ways to accommodate a known combination of environments, platforms, and missions to maintain a known level of performance, and a testing scheme will ensure that various changes maintain capability across the different configurations. This means that reviews and quality checks will be administered for updates, upgrades, and bug fixes to baseline. We also anticipate the implementation of tools to address dependency management, environment, tests, issue tracking, and

creation of issue templates. In addition, this dependency management should support a rapid deployment of foundational operating system and Debian packages.

2) Systematic and modular support for new research (algorithms, platforms, sensors, and environments) through
    a) Communication and collaboration to educate new researchers about how to use and deploy the Stack
    b) Active documentation to understand the provenance of key areas of contribution and licensing
    c) A testing scheme to support cross-team usability during development of a new capability. This testing scheme is preliminary to 1c (above) and facilitates quick transition to testing on developer robots.

This requirement is focused on the Developer role. Some additional documentation and tooling are required to help people in this role because they may not be as comfortable with the code. They might also take some steps in development outside of our tooling and infrastructure. The testing scheme should include verification of code functionality as well as components (data, model, scenarios, etc.) to support published results.

3) A systematic process for integrating research into baseline and changing the baseline through
    a) Active documentation that describes new research: what, how, and why, and limitations
    b) Established acceptance criteria consisting of unit tests, performance tests, preflight checks, and reviews/quality checks
    c) Process for rolling back integrations and returning to previous baseline states. This should support different versions of a wiki (e.g., Phoenix hardware) for when the Stack changes are accompanied by hardware changes, such as switch configurations.

This requirement is essential for maintaining the baseline in instances in which a new code is added for additional features. Developers are the main source of this new code and should provide the acceptance criteria or enough information for maintainers to confirm the acceptance criteria. Developers and maintainers together have the role of ensuring requirement 1, including rolling back updates that cause an unusable state.

4) Systematic and modular support for deployment scenarios through
    a) Active documentation for instantiation and configuration: various platforms, environments, sensors
    b) Easily configurable and testable states including offline or multi-platform
    c) Debugging/analysis/simulation tools

d) Questions and answers (Q/A) about deployment?

This requirement is to ensure that operators have the support to work through various deployments, including the following: searchable documentation to find what is needed, a simple mechanism for updating configuration parameters to match the deployment scenario, and many supporting tools to help troubleshoot issues with deployment. This tooling should support the following: capturing current deployment state, capturing appropriate rosbags, replaying problem areas for troubleshooting, and logging/communication. We anticipate some issues with deployment, such as deploying in areas without internet access. We also anticipate a need for the following tool requirements for rapid reconfiguration:

    a) Simulation tools for pretesting and reanalyzing the configurations
    b) Deployment on multi-platform systems: mission specifications that guide how agents are deployed and evaluated
    c) Q/A about deployment: the documentation will provide examples and allow developers and operators to create and test specified configurations

5) Systematic process for data and model store supporting curation (access, labeling, querying, removing, analysis, issue tracking) and usage through
    a) A standard for data, models, and metadata to be cataloged in the store
    b) A common interface for data and model usage

This requirement covers support for capturing, storing, and removing pertinent data and models. The data can include synthetic data, augmentations of collected data, and experimentally collected rosbags. It will also include metadata to understand what it is and how it should be used, such as platform, calibration, date, and location. Models with appropriate metadata will also be available to support various capabilities and deployments. The definition of a standard and common interface ensures quality and uniformity to the store as well as easy integrations for using the store.

6) Systematic process for managing classification (CUI) and separability of data, models, and code through
    a) Tooling and review of labels and documentation
    b) Validation of classification of the Stack or configuration

This requirement includes the ability to transition quality-of-life features from CUI code to the non-CUI code base. This can include using labels in commits to enable tooling for quick identification of CUI.

1.6 Emergent Requirement

In the context of the users, the following requirements are refined:

1) Tooling for test cards is required to maintain rigorous testing schemes across the range of users:
    a) Simple tools for launching code so that the code and configurations are started the same way for each experiment
    b) Tools/scripts to capture launch files/configs/metadata
2) Preflight check before full deployment:
    a) Unit tests on package dependencies
    b) Check for stderr outputs or build fails
    c) Eliminate some tedious manual checking by maintainers/ease integration by consumers
    d) Check for correct parameters
    e) Auto-bagging with metadata to generate labeled datasets
    f) Check for communications (comms) and cups usage, sensor drivers load

In this section, we explained our goals for research operations, the context of the problem, and the emerging requirements.

2. Approach Outline

In this section, we provide a potential approach to meet the requirements outlined in Section 1. This approach is generated from experience managing Phoenix R1. The relationship between user types and corresponding activities or workflow components are detailed in the Appendix. Workflows Our draft design assumes that these user-activity relationships will remain in implementation.

2.1 Proposed Draft Design

Activities (event/frequency of use):

- **System Documentation:** Manuals, quick start guides, FAQ, key parameters to tune, canonical launch files, Interface Control Documents (ICDs)/diagrams, licensing, git metrics/contributor analysis (Stack catalog), and roadmap that describes major changes in dependencies (ROS version, CUDA, PyTorch, etc.) (continuous)
    - Documentation template autogeneration
    - CUI management tool (labeling)

- **Issue Tracking:** Capturing bugs during building and/or experimentation (event-driven)
    - Potentially, issues arise from Experiment Support, Dependency Management, or Software Management
    - Issue template
    - Post-flight check
- **Test Card Generation:** ARL has a process for planning and conducting robot experiments; however, it should be more automated to have mission specifications and configurations for automated launching. Annotation and analysis should be increasingly automated (before every experiment, demonstration, or regression test)
    - Automated templates
    - Depends on Configuration Management, Experiment Support, informs Software Management, and interfaces to Data Management
- **Configuration Management:** Settings for Stack instantiations (for experiments or specific purposes); all test configurations are probably an aspect of a Test Card (for any simulation or experiment)
    - CUI management tool
    - Aggregated configuration
    - Model/weights management
    - Parameter management
    - Calibration management
    - Deployment of multiple robots (Warthogs and Huskys), also possibly on Jackals and Quadrupeds
    - Post success saving (as opposed to running at each launch or with a new calibration or modification)
    - Simple configurations for consumer role
- **Experiment Support:** Running code on hardware to perform specific missions (for any simulation or experiment)
    - Deployment tools (e.g., docker container building)
    - Preflight check: a "pre-experiment wizard" that applies the mission specification of the test card and checks that 1) all data are available and recording and 2) other variables (CPU, comms bandwidth, driver diagnostics) are consistent.
    - Streamlined interface to robot operations (e.g., a terminal helper tool)
- **Data Management:** Bagging and config captures—"metadata" (with each regression test or field test)
    - Bag extraction and analysis
    - Capture of bag and system states

    - CUI management tool
    - Data archiving
    - Interaction with analysis pipelines
- **Software Management:** Tools for the specific code, which occurs on every MR. This activity also includes versioning and tooling to infer how code changes affect performance. This is a reach goal and may be driven by events such as significant MRs.
    - Updating and rebuilding the Stack
    - Extracting components from external repository to Phoenix R2
    - CUI management tool
    - Policy/guideline enforcement
    - Pushing and tagging
- **Dependency Management:** Environment of the code (executed once per quarter or with major release)
    - Updating/rebuilding system environment
    - Current dependency list tool
    - Feature elimination tool

2.2 Potential Tools

Automating and interfacing these tools will lead to "automated testing," a goal for the cross-cutting theme involving tool design. Tables 2 and 3 describe possible tools to address the requirements for managing robotic hardware and software.

Table 2. Tools to address tasks.

Tool	Requirement	Description/notes
GitLab - Wiki	1.a, 2.a, 3.a, 4.a	Information needs to be easily found and updated, which makes wiki a good place to store documentation for repositories
GitLab - Git	1, 2, 3, 4	Software version control, MR for reviews
GitLab - Container Registry	1.b, 1.c, 2.b, 3.b, 4.b	Stores images for development, testing, and deployment
GitLab - Issues	1, 2, 3, 4, 5, 6	Issue management for tracking priorities and status of current work
Docker Compose	1, 2, 3, 4	Modular way of bringing up a system
dep-scan	1.c, 3.b	Not tested yet: maybe this will help us see the dependencies in a container and used by Python
Fully Automatic Installation	1.d, 4.b	Deployed in the past. We might need a dedicated Debian machine
Jupyter Notebook-created README	1.a, 2.a, 3.a, 4.a, 4.c	Use a Jupyter Notebook to create the readme.md and use the same Notebook to run code in steps, in the README (example)
GitLab - CI/CD	1.b, 1.c, 2.b, 3.b, 4.b	Automation for running tests and checks to ensure quality
rosbag2 transport	5.a	Tool for reading bagged messages from a C++ script rather than through the CLI
Polars	5.a	Python/R/NodeJS-based libraries for accessing and curating structured data
MLflow	2.c, 2.d, 5.a, 5.b	Tools for managing the ML lifecycle: tracking, model registry/management, evaluation, integrations with PyTorch and TensorFlow

Table 3. Requirements/activities mapping.

Requirement	Activities	Description/notes
Baseline state		
1.a - Documentation	System Documentation	Manuals, quick starts, FAQs, ICDs, diagrams, roadmaps, Jupyter Notebooks
1.b - Testing	Test Card Generation, Configuration Management, Experiment Support, Software Management, Dependency Management	Create test cards, setup stacks, run missions, update software, update dependencies
1.c - Reviews	Issue Management, Software Management, Dependency Management	Bug fixes, code reviews, CI/CD pipelines, review software updates, review dependency updates, include notifications about issues developers are having
New research		
2.a - Documentation	System Documentation	Description of new capability and limitations, wiki, Jupyter Notebook README
2.b - Testing	Data Management, Software Management, Dependency Management	Add to testing framework to ensure the systems work as expected
2.c - Data curation	Data Management	Base cases demonstrating research capability
Integrate research		
3.a - Documentation	System Documentation	ICD, dependencies
3.b - Acceptance criteria	Test Card Generation, Experiment Support, Software Management, Dependency Management, Data Management	CI/CD pipelines, performance metrics, preflight checks
Deploy		
4.a - Documentation	System Documentation	Review configuration settings, platforms, environments, sensors, Stack catalog, interface with test cards, configurations, and analysis tools
4.b - Configuration	Configuration Management, Software Management, Dependency Management	Edit configuration settings, edit software versions, edit dependencies, manage calibrations and multi-robot configurations
4.c - Usability tools	Configuration Management, Experiment Support, Data Management	Run experiments and capture bags, Save config, data, metadata, create/ingest simulation data
5 - Data/model store	Data Management	Store, retrieve, delete data and models, label
6 - Classification	All	Maintain sanitation in repository via commits, labels in data and model stores, documentation via reviews

3. Conclusions

After the development of Phoenix R1 and R2 to support intramural and extramural efforts at ARL, SISD has deployed autonomous ground platforms in a diverse set of environments. During this period of development, many collaborators have contributed to software development. In addition, this division has deployed robots in larger groups, in missions over larger distance and time scales. During these experiments and follow-on discussions, we identified and grouped a series of user types and corresponding activities necessary for fielding robotic platforms. This technical note describes a set of corresponding requirements for successfully fielding these platforms and possible approaches for meeting these requirements. Further development and experimentation are required to create the workflows envisioned here.

Appendix. Workflows

This section catalogs the roles and components of workflow.

A.1 Role

A.1.1 Developer

1) Task
    a. New Research
    Developer wants to design or improve on a capability contributing original research
        i. Identify area for contribution
            1. Novel idea
            2. Prior identified deficiency (cataloged anywhere?)
            3. Literature review
        ii. Identify needs to conduct research
            1. Platform
                a. Does research depend on single/multi-robot?
            2. Sensor
                a. Does research depend on a specific type of sensor?
            3. Data
                a. Review already collected data for relevance
                b. Generate new data (capture/simulate)
            4. Model
                a. Review already available models for relevance
                b. Generate new model
        iii. Develop and test capability
            1. Coding
                a. Environment setup (software, operating system)
                b. Framework to tie into (ROS2)
            2. Testing
                a. Launch system/test
                b. Collect metrics/observations/metadata showing publishable results
        iv. Submission of work/paper
            1. Software configuration management
                a. Finished code contribution paper is based on
                b. Data

c. Models
d. Metadata

2. Paper writing
    a. Composition of development effort and supporting material (metrics/observations/ data/metadata/models)

A.1.2 Operator

1) Task
    a. Collect data

    Operator has been tasked with collecting data in a new environment

        i. Choose platform
            1. What sensor/robot platform makes sense for the data collect
        ii. Choose environment
            1. What about the environment needs to be collected, and how do we best collect
        iii. Choose system/software capability
            1. What version of the system needs to run; is that compatible with the current setup; how easy is this to change/update and reconfigure
        iv. Choose data format
            1. How are we collecting and saving data; what metadata needs to be captured

A.2 Activity

A.2.1 System Documentation

1) Task
    a. New capability
        i. Review current documentation and find areas for new contribution
        ii. Add sections describing new capability, including usage requirements
        iii. Edit any other relevant areas for referencing new capability
    b. Upgrading dependencies
        i. Review current documentation for affected dependencies
        ii. Update the documentation including new upgrades and versions of the software affected

A.2.2 Data Management

1) Task
    a. Find all datasets for a particular environment
    b. Find all datasets that use cameras
    c. Add new data
    d. Add new model
    e. Delete data based on deprecated sensor

A.2.3 Fast Setup

1) Task
    a. Setting up a robot (consumer or operator):
        i. across multiple platforms before an experiment
        ii. on a single robot at a field test where the internet may not be available
    b. Setting up test infrastructure:
        i. Create a local git repository
        ii. Create a local docker container server
        iii. Create a local Real-Time Kinematic (RTK) base station that forwards RTCM messages (NTRIP Server)
        iv. Reinitialize firmware
        v. Manage configuration for multiple robots
        vi. Store and train on collected data

The tool comparison is an additional comparison of these software tools with some annotations about where the tools might fit in a workflow, and there are some estimates about how to integrate these tools. These estimates are made with respect to our specific use cases with limited information about how these tools actually work or time investments required.

Table A-1. Tool comparison.

Tool name	Would fulfill requirement	Description/notes	Stand-up time	Learning curve
Gitlab - Git	1, 2, 3, 4	Software version control, merge requests for reviews	1 day	Easy
Gitlab - Wiki	1.a, 2.a, 3.a, 4.a	Information needs to be easily found and updated making wikis a good place to store documentation for repositories	1 week	Easy
Gitlab - Container Registry	1.b, 1.c, 2.b, 3.b, 4.b	Stores images for development, testing, and deployment	1 week–1 month	Medium
Gitlab - Issues	1, 2, 3, 4, 5, 6	Issue management for tracking priorities and status of current work	1 week	Easy
Gitlab - CI/CD	1.b, 1.c, 2.b, 3.b, 4.b	Automation for running tests and checks to ensure quality	1 week–1 month	Medium
Docker Compose	1, 2, 3, 4	Modular way of bringing up a system, including multi-robot systems	1 week–1 month	Medium
dev containers			??	??
dep-scan	1.c, 3.b	Not tested yet: maybe this will help us see dependencies in a container and used by Python	1 week–1 month	Medium
Fully Automatic Installation	1.d, 4.b	Deployed in the past. We might need a dedicated Debian machine	more than 1 month	Easy
Jupyter Notebook - README	1.a, 2.a, 3.a, 4.a, 4.c	Use a Jupyter Notebook to create the readme.md and use the same notebook to run code in steps, in the readme (example)	1 week	Easy
ros2bag transport	5.a	Tool for reading bagged messages from a C++ script rather than through the CLI	1 week–1 month	Medium
Polars	5.a	Python/R/NodeJS-based libraries for accessing and curating structured data	1 week–1 month	Medium
MLflow	2.c, 2.d, 5.a, 5.b	Tools for managing the ML lifecycle: tracking, model registry/management, evaluation, integrations with PyTorch and TensorFlow	more than 1 month	Medium
Django	2.c, 2.d, 5.a, 5.b	Database management, management of metadata	1 week–1 month	Hard
SciKit-Learn	5.a, 5.b	Tools for ML, including classification, regression, model selection, and preprocessing	1 week–1 month	Medium
arl-warthog-sim	1.b.ii, 1.c, 2.c, 3.b, 4.b, 4.c,	Existing Unity simulator for pre-robot testing of new Stack components and regressions	1–2 weeks	Easy
Phoenix GUI	1.a, 2.a, 4.a, 4.b, 4.c, 5?, 6?	GUI for configuring the Stack to run with different available components	1 week–1 month	Medium
CRL LogQS/ARL Reproduction		GUI for browsing bag files with tags and preview	1–2 months	Easy
ansible		Configuration management tool	. . .	. . .
prox-mox		Software configuration and deployment tool	. . .	. . .
Minio	2.c, 5.a	Unstructured data storage suite	. . .	. . .
arl-status-tool	1.b, 3.b, 4.b, 4.c,	Captures metadata on autonomy	. . .	. . .
preflight-check	1.b	Automated robot assessment before testing	. . .	. . .

List of Symbols, Abbreviations, and Acronyms

AAR	after-action report
ARL	Army Research Laboratory
CI/CD	continuous integration and continuous deployment
comms	communications
CPU	central processing unit
CUI	Controlled Unclassified Information
DARPA	Defense Advanced Research Projects Agency
DEVCOM	U.S. Army Combat Capabilities Development Command
DevOps	Development Operations
GVSC	Ground Vehicle Systems Center
ICD	Interface Control Document
ML	machine learning
MR	merge request
NTRIP	Networked Transport of RTCM via Internet Protocol
Q/A	questions and answers
R1	revision 1
R2	revision 2
RCTM	Radio Technical Commission for Maritime
research ops	Research Operations
ROS	Robot Operating System
RTK	Real-Time Kinematic
SISD	Science of Intelligent Systems Division
TB	terabyte

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CI TECH LIB